

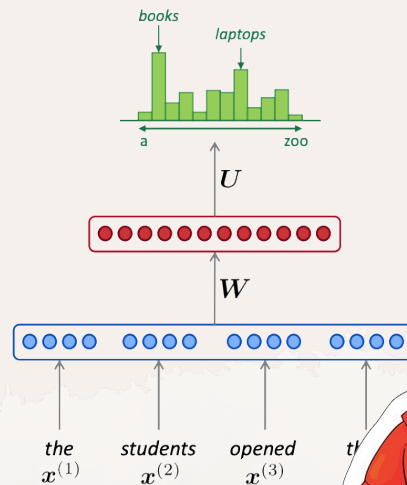
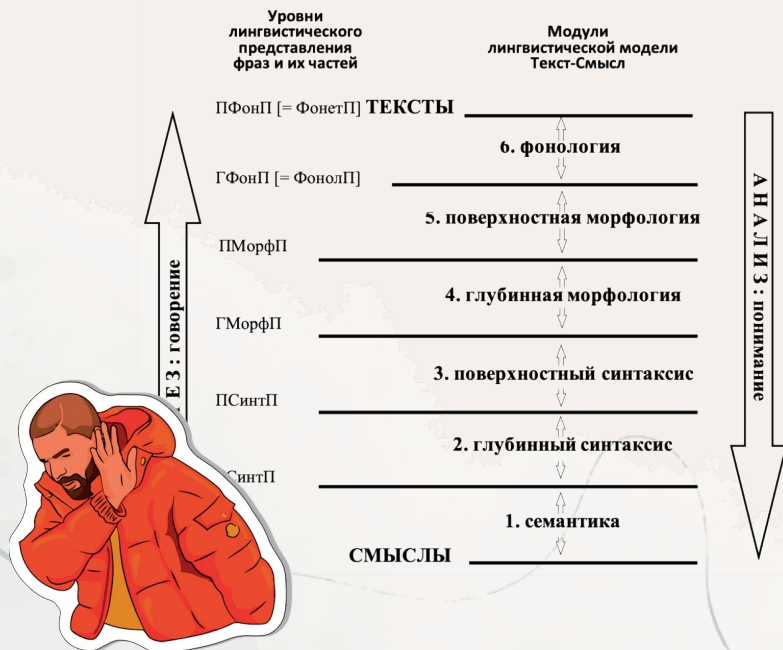
# Языковые модели




# Языковое моделирование

Когда хотим исследовать, как устроено изнутри  
явление, разобрать которое мы не можем,  
строим модели

# Моделирование в КЛ





В компьютерной лингвистике языковая модель – это **вероятностное распределение слов в текстах:**

- насколько вероятно данное наблюдение (последовательность слов) в языке? Например, “съешь еще этих мягких французских булок”. А “дом собака зеленый бегать”?
- зависит от конкретного языка

# Для чего это нужно?

- Для машинного перевода: например, если мы хотим перевести “The human race” на русский, у слова race явно больше одного значения. Но после слова “человеческий” выше вероятность встретить слово “раса”, чем “гонка” (наверное)
- Для задач типа распознавания устной речи: чтобы правильно выбрать между двумя похоже звучащими словами
- Для автоматического исправления орфографических и грамматических ошибок
- Для того, чтобы облегчить вам набор текста на телефоне...

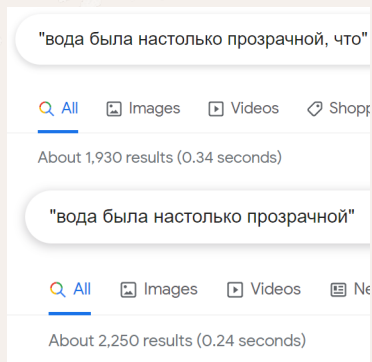
Autocorrect text falls...

Huh?

Fails. Never mind!

# Как посчитать?

- Используем теорию вероятности
- Считаем вероятности на корпусе текстов. Допустим, хотим вычислить вероятность фразы «вода была настолько прозрачной» + «что»
- Нам нужно посмотреть, сколько раз они встретились в нашем корпусе вместе и сколько раз отдельно встретила фраза «вода была настолько прозрачной». Я посмотрю в гугл-поиске:



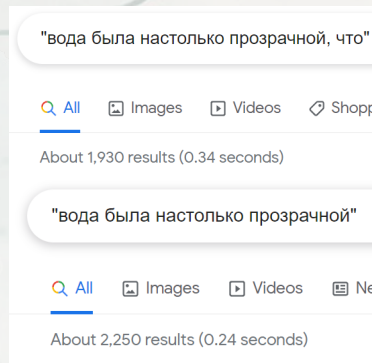
Получилось что-то такое:

$$P(w|h) = \frac{C(\text{вода была настолько прозрачной, что})}{C(\text{вода была настолько прозрачной})} = \frac{1930}{2250} = 0.857$$



$$P(w|h) = \frac{C(\text{вода была настолько прозрачной, что})}{C(\text{вода была настолько прозрачной})} = \frac{1930}{2250} = 0.857$$

Как думаете, в чем проблемы с таким способом подсчета?



The background is a light cream color with abstract, wavy, watercolor-like shapes in pale blue and green. On the right side, there is a faint, yellow outline of a human head in profile, facing left. Inside the head's silhouette, there are two distinct clusters of small, overlapping blue circles, resembling a brain or neural network. Other smaller blue circles and shapes are scattered throughout the background.

01

# N-gram models



# Цепное правило вероятности

- На самом деле:

$P(\text{что}|\text{вода была настолько прозрачной})$

$$= \frac{C(\text{прозрачной, что})}{C(\text{прозрачной})} * \frac{C(\text{настолько прозрачной})}{C(\text{настолько})} * \dots$$

- То есть, вероятность встретить слово «была» после слова «вода» должна учитываться при подсчете вероятности встретить слово «настолько» после слов «вода была» и так далее
- Это называется цепное правило вероятности (вероятность одновременности нескольких событий равна произведению их вероятностей)
- Но для очень длинного предложения нам придется ужасно много всего вычислить!

# Марковское свойство

- Предположим, что вероятность нашего текущего слова определяется только предыдущим словом
- Это называется марковское свойство (по имени русского математика А.А. Маркова), или свойство отсутствия памяти
- Модели Маркова – такие вероятностные модели, которые предполагают, что мы можем предсказывать будущее, не заглядывая слишком далеко в прошлое
- Если мы смотрим только на предыдущее слово, у нас получается **биграм** (биграмная модель).
- Можем обобщить и смотреть на два предыдущих слова (тогда будет **триграм**)
- Или на N предыдущих слов (**N-грам**)

$$P(\text{future} \mid \text{present, past}) = P(\text{future} \mid \text{present, } \text{my})$$

Markov property ↗

# Функция правдоподобия

- Получается, чтобы смоделировать весь язык, нам нужно вычислить все вероятности всех слов
- Все эти вероятности вместе представляют собой вероятностное распределение (joint probability distribution)
- Собственно говоря, функция правдоподобия – это и есть такое вероятностное распределение, описанное в виде математической функции
- В этой функции наши вероятности слов – это параметры
- Итак, чтобы смоделировать язык, нужно подобрать параметры таким образом, чтобы наша математическая функция выдавала что-то максимально похожее на настоящий язык

# Метод максимального правдоподобия

Как же вообще рассчитать эти самые вероятности слов, они же – параметры функции правдоподобия?

Для этого используем такой способ:

- Чтобы вычислить вероятность конкретного биграма для слов  $w_{n-1}, w_n$ , посчитаем частоту этого конкретного биграма, а потом поделим на суммы частот всех биграмов, у которых первое слово – наше  $w_{n-1}$ :

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

- Эту формулу на самом деле можно упростить: ведь количество всех биграмов с первым словом  $w_{n-1}$  просто равно частоте самого слова  $w_{n-1}$ .

# Метод максимального правдоподобия

- Получается, что каждый раз мы вычисляем частоту в корпусе нашего биграма и делим ее на частоту первого его слова.
- Это отношение называется **относительная частота** (relative frequency)
- Относительные частоты максимизируют правдоподобие корпуса, на котором мы их вычисляли, для нашей языковой модели
- Допустим, у нас есть корпус на миллион слов, и слово «китайский» встречается в нем 400 раз.
- Вероятность того, что случайно выбранное слово в этом корпусе окажется «китайский», равна  $400 / 10^6 = 0.0004$ .
- В другом корпусе эта вероятность может быть другой, но в корпусе такого размера, как наш – это наилучшая вероятность

# Практические особенности

- Биграмы используют довольно редко, на практике, конечно, чаще встречаются 5- или 6-граммы
- Обычно вероятности записываются не as is, а как логарифмы. Это связано с тем, что вероятности могут быть маленькими числами, а когда начинаем их умножать, они еще уменьшаются. Логарифмирование все делает немножко покрупнее.
- К тому же, логарифмы нужно складывать вместо умножения
- Поэтому говорят о maximum log-likelihood estimation



# Генерация текста

Модель может пытаться породить текст на основании обучающего корпуса! (Выбирает случайным образом из  $n$  самых вероятных граммов в зависимости от того, что было предыдущим)

Например, обученная на учебнике «Введение в общий синтаксис» триграм-модель порождает:

принципы не отражают потребностей коммуникации п . с том для действующего лица , времени и в таких случаях требуется правило другого рода , а не терминальных , а второе зависимое местоимение самая изменяется и соответственно той мере усилий , которая может быть извлечена из универсальных принципов



# Evaluation

02

# Как оценить качество модели?

- Самое очевидное – проверить на конкретной задаче (мы же создаем языковые модели для практического применения?)
- Это называется **extrinsic evaluation** (внешняя оценка)
- Но есть и **intrinsic evaluation** (каждый раз гонять на практических задачах долго и дорого)
- Для внутренней оценки нам необходим тестовый корпус (которого модель не видела)
- А потом нужно оценить, насколько высокую вероятность модель приписывает тестовому корпусу (он ведь существует, а значит, вероятен...)
- Обязательное условие – чтобы в тестовом корпусе не было ни одного предложения, которое было в обучающем

# Перплексия

- Обычно для оценки модели используют не вероятность, а перплексию (степень неопределенности вероятностной модели)
- Перплексия модели – это обратная вероятность тестового корпуса, нормированная по количеству слов:

$$\text{perplexity}(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

- (К этой жуткой формуле тоже применяется цепное правило)
- Получается, раз это обратная вероятность, то чем выше вероятность предложения в тестовом корпусе, тем ниже перплексия. Мы ее **минимизируем**

# Проблемы

- Из-за этих проблем у нас может возникать ситуация деления на ноль
- Бороться с ней помогает сглаживание: просто приплюсовываем ко всем знаменателям какое-нибудь очень маленькое число
- Несловарные слова можно заменять на специальный токен <UNK>

## Есть две вещи:

### 1. Разреженность

Модель зависит от обучающего корпуса. Каким бы большим он ни был, все равно найдутся n-граммы, которые в самом деле ок, но их в нем нет или очень мало

### 2. OOV-слова

Неологизмы, окказионализмы, опечатки – как тебе такое, Илон Маск?

# Проблемы

- Что делать с редкими  $n$ -граммами?
- Допустим, если мы оцениваем вероятности триграмов, и для какого-нибудь у нас просто нет примеров, мы можем попробовать откатиться к биграмам. Этот метод называется backoff. Он применяется только тогда, когда у нашего  $n$ -грама частота равна 0
- А можно в принципе оценивать вероятности  $n-1$ -грамов: этот метод называется интерполяция.
- То есть, для триграма мы просто оцениваем вероятности самого триграма, биграмов и униграмов, из которых он складывается, а потом вычисляем взвешенную сумму (взвешенная – с коэффициентами, чтобы в сумме они давали единицу)





03

## word2vec и дистрибутивная семантика

# One Hot Encoding

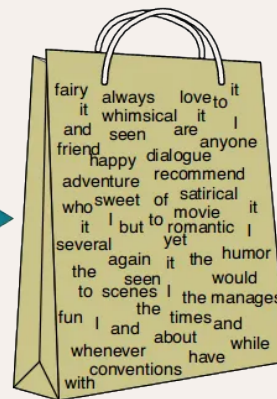
- Допустим, хотим скормить текст нейронной сети или классическому алгоритму машинного обучения
- Но машина понимает только цифры
- Как можно представить слова в виде чисел?
- Такой способ называется One Hot Encoding
- Хорошо, но неинформативно

	a	cat	is	this	...
this →	0	0	0	1	...
is →	0	0	1	0	...
a →	1	0	0	0	...
cat →	0	1	0	0	...
					⋮

# BoW

- Мы можем хотеть представить в виде чисел сразу весь текст
- Тогда можно собрать все слова в словарь, пронумеровать и представить текст как частоты наших слов
- Это уже поинформативнее, правда, мы получаем информацию только про текст, но не про отдельные слова


I love this movie! It's sweet,  
but with satirical humor. The  
dialogue is great and the  
adventure scenes are fun...  
It manages to be whimsical  
and romantic while laughing  
at the conventions of the  
fairy tale genre. I would  
recommend it to just about  
anyone. I've seen it several  
times, and I'm always happy  
to see it again whenever I  
have a friend who hasn't  
seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...



# TF-IDF

- Еще более хитрый способ представить текст в виде чисел – использовать не просто частоты слов в тексте, а их TF-IDF (term frequency – inverted document frequency), которая будет учитывать еще и важность слова для конкретного документа
  - Term frequency (частота слова) – отношение частоты слова к общему числу всех слов
  - IDF (обратная частота документа) – инверсия частоты, с которой слово встречается в разных документах
- 

# TF-IDF

$$\text{TF: } \frac{\text{частота слова}}{\text{все слова в корпусе}}$$

$$\text{IDF: } \log \frac{\text{количество документов в корпусе}}{\text{количество документов, в которых встречается наше слово}}$$

Пример:

у нас в корпусе 3 документа, в которых в сумме 100 слов.

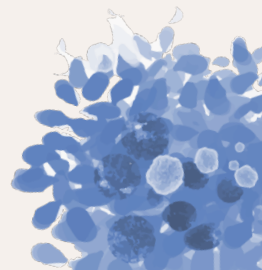
слово «котик» встречается 30 раз, но в двух документах.

$$\text{его TF-IDF} = \frac{30}{100} \cdot \log \frac{3}{2} = 0.12$$

а слово «карбюратор» встречается 15 раз, но только в одном документе.

$$\text{его TF-IDF} = \frac{15}{100} \cdot \log \frac{3}{1} = 0.16$$


- BoW и TF-IDF действительно работают, когда нам нужно, например, классифицировать или кластеризовать тексты
- Но о словах мы по-прежнему ничего не знаем
- Рассмотрим два примера:
  - «По улице ездят машины»
  - «Машина вычислила perplexity для языковой модели»
- В этих предложениях слово «машина» явно употребляется в разных значениях. Как мы это поняли?





# Дистрибутивная гипотеза

- В 1954 году Зеллиг Харрис выдвинул гипотезу о том, что если два слова часто встречаются в похожих контекстах, их значения похожи
- Например, рассмотрим такие контексты:
  - «Вася часто пьет ...».
  - «... – алкогольный напиток».
  - «Продажа ... после 11 вечера запрещена».
- Какие варианты подходят сюда?



**Дистрибутивная гипотеза**  
утверждает, что статистическое  
распределение лингвистических  
элементов в контексте определяет  
их семантическое поведение

# Слова как числа

- Давайте попробуем учесть контексты, в которых встречаются слова, когда будем превращать слова в числа:
- Возьмем большой корпус текстов (обучающий датасет)
- Выберем окно контекста (будем смотреть  $n$  слов слева и справа от нашего слова)
- Пройдемся по всему корпусу и посчитаем частоты всех слов, которые попали в наше окно контекста
- Получится **вектор**
- Этот вектор будет  $V$ -мерным, если наш словарь всех уникальных слов имеет объем  $V$
- Такой вектор называется **эмбединг**

# Косинусная близость

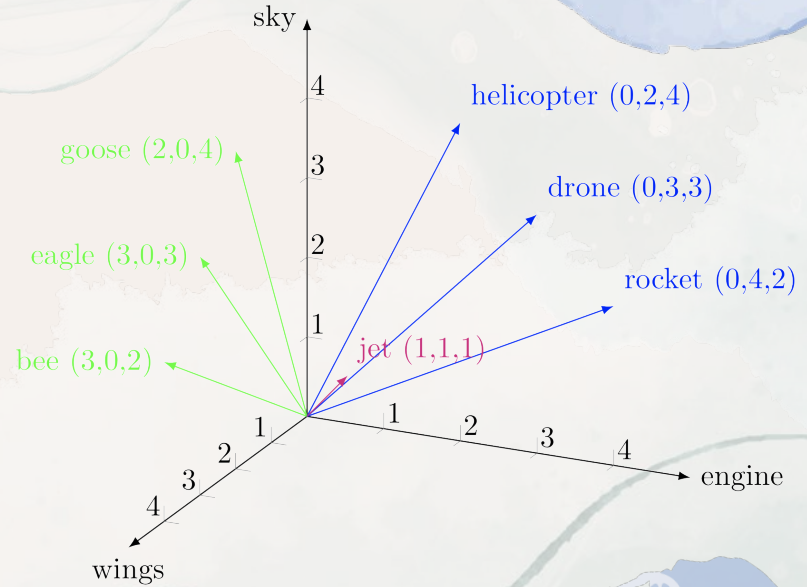
- Если мы представляем все слова в виде векторов, то векторы имеют направление в пространстве
- Бывают векторы, которые смотрят примерно в одну сторону, мы считаем, что они похожи
- Можно вычислить похожесть векторов, посчитав косинусную близость:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

- То есть, нужно скалярное произведение векторов A и B поделить на произведение их норм (не пугайтесь, все это умеет делать, например, библиотека питона `scipy`)

# Косинусная близость

- Если мы собрали наши эмбединги с помощью контекстов слов, то окажется, что они располагаются соответствующим образом:



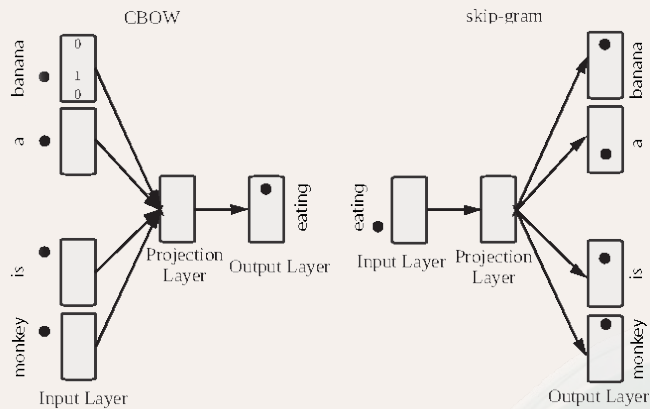
# word2vec

- В 2013 году Томас Миколов придумал, как вычислять эмбединги с помощью нейронных сетей
- Его эмбединги получили название word2vec
- Они еще называются статическими: потому что для каждого слова в словаре может быть только один эмбединг, следовательно, «коса» как прическа и как инструмент все равно получат одни и те же цифры
- Основная идея: нейронная сеть учится предсказывать слово по контексту. На вход она получает ОНЕ слов и подбирает свои коэффициенты таким образом, чтобы лучше решать свою задачу
- Задача может быть обратная: предсказывать контекст по слову



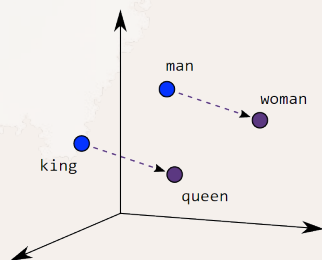
# word2vec

- Эти две задачи называются Continuous Bag of Words и skip-gram
- Skip-gram кажется менее интуитивным, но работает обычно лучше

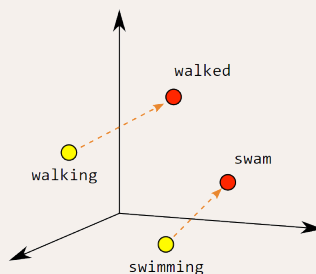


# word2vec

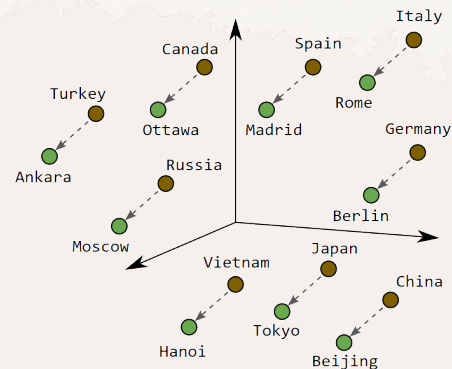
- Такие эмбединги настолько хорошо собирают семантическую информацию о слове, что отношения между ними (иногда) передают и отношения между словами:



Male-Female



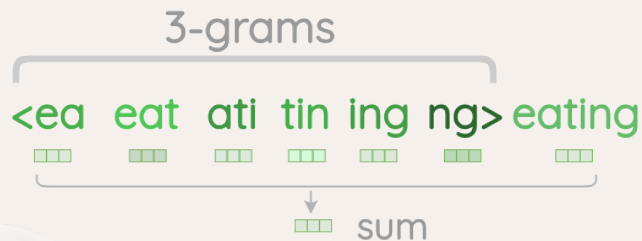
Verb Tense



Country-Capital

# word2vec & OOV

- Та же самая история с неизвестными словами: что с ними делать?
- Придумали делить слова дополнительно на кусочки: если у нас есть эмбединги для частей слова, например, мы знаем часть слова «енок» (как в «котенок»), то «бокренок» для нас уже не будет чем-то совершенно неизвестным
- Этот вариант называется **fasttext**



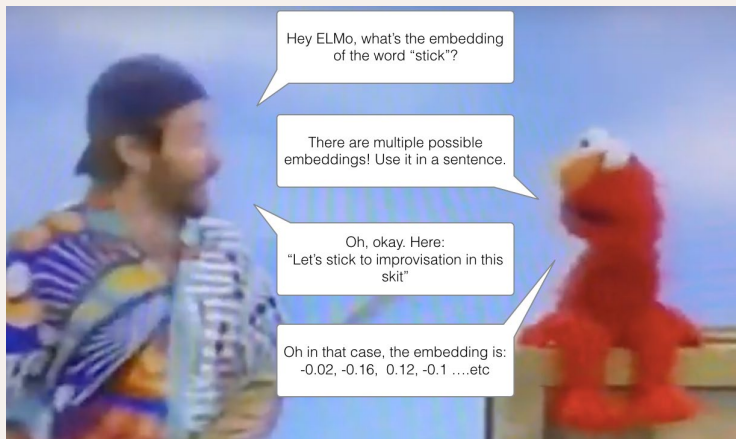


04

# Contextualized embeddings

# RNN и ELMo

- Вспоминаем: RNN и LSTM архитектуры умеют обращать внимание на контекст – ведь они работают с последовательностями
- Можно обучить эмбединги для наших слов на LSTM
- Получим эмбединги, которые зависят от контекста:



# BERT

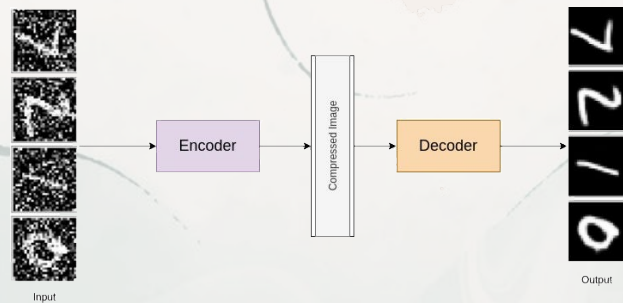
- Но еще лучше с контекстом работают трансформеры: благодаря механизму самовнимания они прямо заточены на контекст
- BERT – это модель-трансформер, которую учили решать задачу **Masked Language Modelling**





# Masked Language Modelling

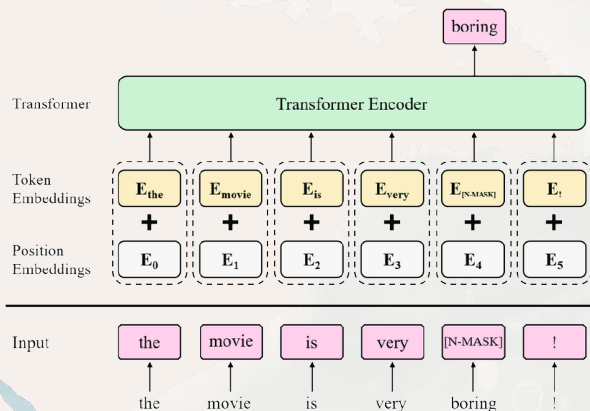
- Можно научить нейронную сеть восстанавливать исходный объект из слегка подпорченного
- Это на удивление хорошо помогает вычлнить важные признаки объекта
- Давайте сделаем то же самое с языком...





# Masked Language Modelling

- Замаскируем некоторые слова в тексте специальными токенами <MASK>
- И заставим наш трансформер решать задачу восстановления замаскированных слов
- Получается, трансформер тоже ориентируется на контекст
- И для каждого слова будет свой эмбединг в зависимости от контекста в каждом конкретном случае



# GPT



- GPT – тоже сеть-трансформер
- Но задача, которой она обучается, другая: это просто предсказание следующего слова

Именно поэтому BERT хорошо решает задачи анализа текстов, а GPT хорошо генерирует новые тексты